On Energy-efficient Checkpointing in High-throughput Cycle-stealing Distributed Systems

Matthew Forshaw[1], A. Stephen McGough[2] and Nigel Thomas[1]
[1]*School of Computing Science, Newcastle University, Newcastle, U.K.*
[2]*School of Engineering and Computing Sciences, Durham University, Durham, U.K.*

Keywords: Energy Efficiency, Checkpointing, Migration, Fault Tolerance, Desktop Grids.

Abstract: Checkpointing is a fault-tolerance mechanism commonly used in High Throughput Computing (HTC) environments to allow the execution of long-running computational tasks on compute resources subject to hardware and software failures and interruptions from resource owners. With increasing scrutiny of the energy consumption of IT infrastructures, it is important to understand the impact of checkpointing on the energy consumption of HTC environments. In this paper we demonstrate through trace-driven simulation on real-world datasets that existing checkpointing strategies are inadequate at maintaining an acceptable level of energy consumption whilst reducing the makespan of tasks. Furthermore, we identify factors important in deciding whether to employ checkpointing within an HTC environment, and propose novel strategies to curtail the energy consumption of checkpointing approaches.

1 INTRODUCTION

The issue of performance and reliability in cluster computing have been studied extensively over many years (Jarvis et al., 2004), resulting in techniques to improve these properties. The issue of cluster *'performability'* is relatively well understood, but until recently few have considered its consequences for energy consumption.

High-throughput cycle stealing distributed systems such as HTCondor (Litzkow et al., 1998) and BOINC (Anderson, 2004) allow organisations to leverage spare capacity on existing infrastructure to undertake valuable computation. These High Throughput Computing (HTC) systems are frequently used to execute long-running computational tasks, so are susceptible to interruption due to hardware and software failures. Furthermore, in our context of an institutional *'multi-use'* cluster comprising student cluster machines, jobs may also be interrupted by the arrival of interactive users to cluster workstations.

Checkpointing is a fault-tolerance mechanism commonly used to increase reliability by periodically storing snapshots of application state. These snapshots may then be used to resume execution in the event of a failure, reducing wasted execution time. Checkpointing has previously been employed on HTC clusters with little consideration of the energy consumption incurred by checkpointing overheads.

In recent years attention has turned to the energy consumption of IT infrastructures within organisations. Aggressive power management policies are often employed to reduce the energy impact of institutional clusters, but in doing so these policies severely restrict the computational resources available for research computing.

We demonstrate through trace-driven simulation using real-world datasets (Section 2) the detrimental effect of existing checkpointing policies on energy consumption (Section 3), motivating the need for an increased understanding of the impact of checkpointing strategies within HTC clusters. Finally we discuss key considerations when adopting checkpointing in HTC clusters and go further to highlight possible future directions for more energy-efficient checkpointing (Section 4).

1.1 Related Work

Previous works in energy-aware checkpointing have primarily focused on real-time systems (Zhang and Chakrabarty, 2003; Unsal et al., 2002; Melhem et al., 2004) subject to strict energy budgets and deadlines.

More recently, research has sought to understand the overheads and energy implications of fault tolerance mechanisms, including checkpointing, in an-

Forshaw M., McGough A. and Thomas N..
On Energy-efficient Checkpointing in High-throughput Cycle-stealing Distributed Systems.
DOI: 10.5220/0004958302620267
In *Proceedings of the 3rd International Conference on Smart Grids and Green IT Systems* (SMARTGREENS-2014), pages 262-267
ISBN: 978-989-758-025-3

ticipation of exascale High-Performance Computing (HPC). At exascale, increased frequency of faults are anticipated and energy consumption is a key issue (Cappello et al., 2009). To this end, Diouri *et al.* explore the energy consumption impact of uncoordinated and coordinated checkpointing protocols on an MPI HPC workload (El Mehdi Diouri et al., 2012), while Mills *et al.* demonstrate energy savings by applying Dynamic Voltage and Frequency Scaling (DVFS) during checkpointing (Mills et al., 2013).

The application of checkpointing in Fine-Grained Cycle Sharing (FGCS) systems is explored extensively in (Ren et al., 2007; Bouguerra et al., 2011), though without consideration for its implications for energy consumption. In (Aupy et al., 2013), energy-aware checkpointing strategies are investigated in the context of arbitrarily divisible tasks.

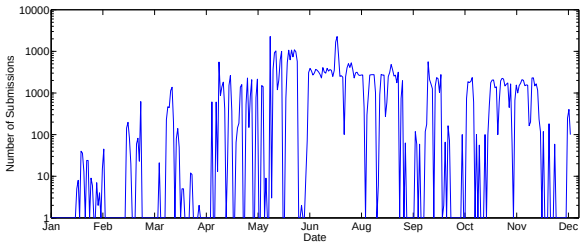

Figure 1: HTCondor job submissions.

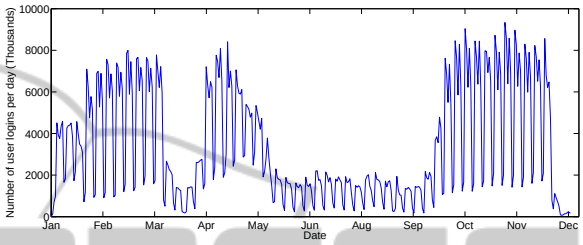

Figure 2: Interactive user arrivals.

2 EXPERIMENTATION

In this paper, we evaluate the efficacy of existing checkpointing schemes using trace-driven simulation on a real dataset collected during 2010 at Newcastle University (McGough et al., 2011), comprising details of all job submissions to Newcastle University's HTCondor (Litzkow et al., 1998) cluster and interactive user activity for the twelve month period.

In 2010, the Newcastle University HTCondor cluster comprised 1,359 machines from 35 computer clusters. The opening hours of these clusters varied, with some respecting office hours, and others available for use 24 hours a day. Clusters may belong to a particular department within the University and serve a particular subset of users, or may be part of a common area such as the University Library or Students' Union building.

Figure 1 shows all HTCondor job submissions for 2010. To aid clarity, the figure is clipped on 3rd June 2010 which featured ˜93,000 job submissions. Figure 2 shows the seasonal nature of interactive user activity within these clusters, demonstrating clear differences between weekends and weekdays, as well as term-time and holiday usage.

2.1 Checkpointing and Failure Model

Choi *et al.* (Choi et al., 2004) present a classification of two types of failures encountered on desktop grid environments; *volatility failures* including machine crashes and unavailability due to network issues, and *interference failures* arising from the volunteer nature of the resources. It is these *interference failures* which we consider throughout this work. Furthermore, we consider resource volatility in the form of scheduled nightly reboots for maintenance.

Figure 3 shows the state transition diagram for the execution of a single job in our system in the presence of these failures. Our checkpoint model differs from those presented in the literature because we assume interruptions may occur during checkpointing.

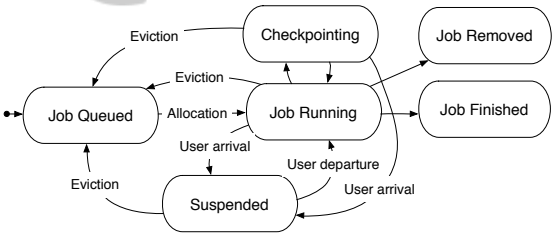

Figure 3: Job state transition diagram.

2.2 Policies

In these preliminary experiments, we evaluate the following three checkpointing strategies:

NONE. This policy represents the policy enacted during 2010 in the Newcastle University HTCondor pool, where no jobs were checkpointed.

C(*n*): Each job is checkpointed every *n* minutes. Hourly checkpointing (**C**(*60*)) is frequently considered in the literature and the HTCondor default strategy equates to **C**(*180*) (UW-Madison, 2013).

OPT. An optimal checkpointing strategy for best case comparison, whereby jobs are checkpointed immediately prior to eviction.

3 RESULTS

Figure 4 shows the mean job overhead under each policy, while Figure 5 shows the impact of these policies on energy consumption. The results shown are mean values taken from twenty simulation runs, with error bars signifying ±1SD. While checkpoint is effective in curtailing wasted execution for long-running tasks, our experimentation finds significant overheads incurred by the checkpointing of short-running tasks unlikely to face interruption. These overheads prolong execution and have a detrimental impact on overall energy consumption.

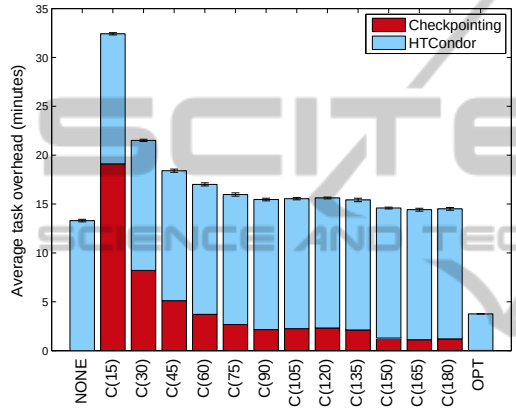

Figure 4: Average Task Overheads.

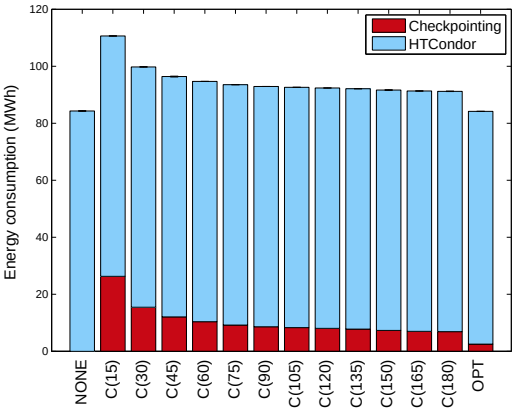

Figure 5: Energy Consumption.

4 DISCUSSION

In this section, we outline the considerations the administrator of an HTC cluster should make when deciding whether to employ a checkpointing mechanism within their environment. Furthermore, we highlight a number of areas of research interest, both with respect to energy-efficient checkpointing generally, and also issues specific to the application of these approaches in the context of multi-use clusters.

Operating Policies. FGCS systems are typically configured to operate conservatively, with the interactive user of a machine given priority over the HTC workload running on the machine. Historically there was significant potential of interference from an HTC job, degrading performance and resposiveness for interactive users of a system. However, now in multicore systems, and with the additional separation afforded by virtualisation technologies, the impact of HTC workloads on interactive users has been shown to be negligible (Li et al., 2009). Relaxing operational constrains preventing HTC jobs from running on resources with interactive users not only increases the capacity and throughput of the system, but also offers significant reduction in energy consumption.

Workload. The efficacy of checkpointing is largely dependent on cluster workload. Checkpointing is most useful when the execution time of a large proportion of the workload exceeds typical resource mean time to failure (MTTF) or user inter-arrival durations, increasing the likelihood of interruption. Furthermore, some jobs do not support checkpointing, while others are unsuitable for checkpointing e.g. those with particularly large application states.

User Base. The Newcastle University HTC cluster supports a diverse user base, from experienced system administrators and Computer Scientists interacting directly with the system, to scientists leveraging its capabilities through user interfaces or submission mechanisms provided to them. Consequently there is a need for checkpointing mechanisms to be transparent and not require in-depth understanding of HTC or programming ability for users to benefit.

Resource Composition. Modern HTC clusters commonly comprise both volunteer and dedicated resources, and increasingly leverage Cloud resources to handle peak loads and offer runtime environments not supported locally. The composition of a cluster is an important factor in determining whether checkpoint mechanisms should be employed. In clusters solely relying on volunteer resources, checkpointing offers an attractive means to deliver favourable makespan in the presence of interruptions. As the proportion of dedicated resources increase, similar benefits may be sought by steering longer-running jobs to these more reliable resources. The implications of checkpointing on workloads running on Cloud resources has not previously been investigated in the literature, but data transfer/storage and instance costs will exacerbate the impact of any checkpoint overheads.

4.1 Checkpointing Support

In determining the benefit of employing a checkpointing mechanism within a cluster, it is important to understand the proportion of the workload and compute resources who support checkpointing. A number of barriers currently exist including operating system dependence of checkpoint frameworks and the requirement to re-compile or re-link executables.

At present, the HTCondor transparent process checkpoint/migration is not supported under Windows. The issue of operating system dependence is often exacerbated in institutional multi-use clusters, with workstations provisioned for the needs of the primary (interactive) users of the system. Students generally demand Windows-based machines so the proportion of resources capable of checkpointing is limited. At Newcastle University, Linux-based machines constitute only ˜*5%* of resources available to HTCondor.

Overcoming this operating system dependence presently relies on application- or user-level checkpointing. These offer greater checkpoint portability and allow checkpoint operations to be conducted at times when application state is smallest. However, these mechanisms assume expert knowledge and often requires access to original source code. Here we propose two approaches to improve checkpointing support while maintaining user-transparency.

Virtualisation. HTC jobs could be executed within a virtual machine on a worker node, providing support for VM- or process-level checkpointing. However, this approach has been shown to prolong execution time for HTC jobs by between *11.7%* and *22.3%* (Li et al., 2009), which combined with increased resource utilisation will increase energy consumption. Furthermore, in VM-level checkpointing, larger snapshots will lead to increased overheads.

Dual-boot Clusters. Booting into a Linux environment would enable support for Kernel-level checkpointing. The use of dual-boot clusters has been considered in terms of HPC clusters (Liang et al., 2012), but its application in multi-use clusters presents additional considerations required to maintain interactive user quality of experience (QoE). The time required to boot between operating systems is likely to be prohibitively long for use during periods with short user inter-arrival durations, though it may prove effective during quiet periods or when clusters are closed for public use. This approach presents the additional benefit of increasing the flexibility of the HTC cluster, offering increased support for Linux jobs which may otherwise have required dedicated or cloud resources.

4.2 Reducing ‘Wasted’ Checkpoints

Through our experimentation in Section 2, we have shown conventional checkpointing policies to be inadequate in reducing the makespan of tasks while maintaining acceptable levels of energy consumption. The predominant factor in the prolonged execution of some tasks and increased energy consumption incurred by these approaches is ‘wasted’ checkpoint overheads, the overhead of checkpoints which are not subsequently requested for recovery. We hope to mitigate these effects by intelligently identifying particular jobs and opportune moments at which checkpoint operations are likely to be beneficial.

Short-running jobs are less likely to be impacted by failures, with checkpointing overheads more likely to increase makespan for these jobs. While execution time is not known *a priori* and user estimates have been shown to be inaccurate (Bailey Lee et al., 2005), we may estimate the execution time of jobs belonging to a batch based on the execution time of other (ongoing or completed) jobs. This leads to the potential for adaptive checkpointing strategies considering expected runtime of jobs.

If we are able to measure the probability of interference from interactive users, we may design checkpointing and resource allocation strategies to mitigate such failures. While we have previously shown the interactive user workload to be accurately forecastable (Bradley et al., 2013), we hope to achieve comparable results through intuitive policies leveraging system knowledge. For example, adaptive checkpoint intervals may be applied depending whether the cluster is open or closed for use by interactive users. Also in the case of departmental clusters used for teaching and practical lab sessions, the central University timetabling system could be used to inform checkpointing and resource allocation decisions, avoiding allocating jobs to a cluster with scheduled sessions approaching, and checkpointing jobs before these scheduled sessions commence.

4.3 Energy-aware Checkpoint Storage

Typical checkpoint schemes in the literature assume nodes acting as centralised checkpoint repositories for all tasks in the system. This relies on the availability of dedicated infrastructure for the purpose, and represents a central point of failure and performance bottleneck. Furthermore, these centralised checkpoint repositories constitute a significant baseline energy load, impacting the energy proportionality (Barroso and Holzle, 2007) of the system as a whole.

The energy cost of centralised checkpoint repos-

itories may be reduced through the use of energy-aware server provisioning to power off repository nodes during periods of low utilisation to save energy. This dynamic scalability introduces a policy decision surrounding the trade-off between worker- and repository-side energy consumption and checkpoint availability. By reducing the frequency of checkpoints taken by the system, it may be possible to allow a checkpointing repository to transition into a low-power state, but in the event of failures, the repeated computation may be more costly in terms of energy than if the checkpoint repository has remained powered up. Furthermore, powering down checkpointing nodes would make the system more susceptible to failures of the remaining checkpoint nodes.

Alternatively, worker nodes performing computation could operate as checkpoint repositories, storing checkpoints for other jobs. While High-Performance Computing (HPC) workloads such as MPI-based parallel applications rely on low-latency interconnects between nodes, HTC jobs typically have minimal network requirements so we expect the impact on the resident job to be negligible. The energy consumption of a given node is dominated by the CPU consumption on the resident job, with only a small proportion of its dynamic power range attributed to system memory and network subsystems, making this manner of checkpoint storage cheap in terms of energy consumption. However, unlike centralised checkpoint repositories which are assumed to be available except for machine or software failure, these resources would be subject to interruptions by interactive users, raising a number of key policy decisions:

Node Selection. In addition to the energy consumption of a node, transfer time is an important factor in checkpointing performance. Network costs are lower to transfer checkpoints to machines within the same cluster, but strong inter-cluster correlation of machine availability increases the likelihood that these checkpoints will subsequently be evicted.

Checkpoint Replication. Whether dealing with volatility failures on dedicated resources, or interruptions from interactive users, checkpoint replication is required to ensure availability of checkpoints. There exists a trade-off between the cost of replication and checkpoint availability. Storing too many replicas incurs an overhead and energy cost, while insufficient replication leads to repeated computation.

Retention. As checkpoints age, the benefits of their use for recovery diminishes, hence, there is a need for a mechanism to curtail the retention of outdated checkpoints. In an uncoordinated approach, checkpoint retention is managed through the use of a retention interval, after which checkpoints are discarded. In a coordinated approach, a checkpoint repository is informed when a job completes or leaves the system, or subsequent checkpoints are produced, signalling that its checkpoints may safely be removed. This requires additional communication between the system and checkpoint repositories, though offers potential benefits through multi-version checkpointing.

4.4 Energy-aware Proactive Migration

In addition to enabling recovery from failures, checkpointing mechanisms may also be used to support proactive migration of computational tasks to reduce makespan and energy consumption. Examples of such migrations include:

1. Migrate a task to a more computationally powerful resource to reduce execution time. These more powerful machines are typically newer and more energy efficient, leading to further energy savings.
2. Migrate a task to a more energy-efficient resource to reduce energy consumption.
3. Migrate a task to a quieter resource to reduce the likelihood of job eviction by interactive users.
4. Migrate a task to avoid scheduled interruptions, e.g. all campus computers at Newcastle University reboot daily between 3am and 5am to perform routine maintenance and apply security updates.

In each instance, the cost of the migration operation must be balanced against the potential benefits to determine whether a migration is viable. These migration policies are not mutually exclusive, and we anticipate combining these will yield the greatest benefit.

4.5 Impact on Policy Decisions

The introduction of a checkpointing mechanism introduces an interesting interplay between a number of existing policy decisions within an HTC cluster.

Resource Allocation. In (McGough et al., 2013) we introduce resource allocation strategies to minimise the likelihood of job eviction and reduce energy consumption. The introduction of checkpointing provides opportunities to develop novel checkpoint-aware resource allocation strategies. For example, in HTC clusters where only a subset of resources or jobs support checkpointing, wasted execution can be reduced by allocating longer-running jobs to resources which support checkpointing and non-checkpointable jobs to quieter resources. Resource allocation should also consider data locality when resuming larger jobs, selecting resources with lowest checkpoint transfer cost. Furthermore, in situations where a given job

may only run on a particular subset of resources, it would be beneficial to store checkpoints on or close to resources capable of resuming its execution.

Replication. Replication of jobs in an HTC system is generally dismissed due to increased overheads and reduced system throughput. While this holds true for heavily utilised HTC clusters, there is a case for energy-conscious replication of jobs. The Newcastle University HTC cluster features significant spare capacity so the replication of certain jobs need not impact the makespan of other jobs. If replicas were to run alongside interactive users, the energy cost associated with the HTC workload would also be minimal.

Vacation. Furthering the desire to minimise the impact of HTC workloads on interactive users of computers, HTC clusters are configured to ensure an interrupted job vacates a resource quickly. Many clusters including Newcastle are configured to vacate HTC tasks immediately without checkpointing, leading to wasted execution. A more beneficial approach would be to allow checkpoint at the time of vacation, but limit the impact on users with a timeout interval after which the checkpoint operation is abandoned.

5 CONCLUSION

In this paper we have shown existing checkpointing mechanisms to be inadequate in reducing makespan while maintaining acceptable levels of energy consumption in multi-use clusters with interactive user interruptions. Our preliminary experimentation shows the naive application of checkpointing approaches to have the potential to negatively impact energy consumption, but small changes to make these strategies energy- and load-aware may lead to significant benefits. We highlight key considerations when adopting checkpointing in an HTC cluster and motivate a number of areas of future research interest in energy-efficient checkpointing. A detailed evaluation of new energy-aware checkpointing strategies will form the basis of our ongoing research.

REFERENCES

Anderson, D. P. (2004). Boinc: A system for public-resource computing and storage. GRID '04, pages 4–10.

Aupy, G., Benoit, A., Melhem, R. G., Renaud-Goud, P., and Robert, Y. (2013). Energy-aware checkpointing of divisible tasks with soft or hard deadlines. *CoRR*, abs/1302.3720.

Bailey Lee, C., Schwartzman, Y., Hardy, J., and Snavely, A. (2005). Are user runtime estimates inherently inaccurate? volume 3277 of *LNCS*, pages 253–263.

Barroso, L. and Holzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12):33–37.

Bouguerra, M., Kondo, D., and Trystram, D. (2011). On the Scheduling of Checkpoints in Desktop Grids. CCGrid '13, pages 305–313.

Bradley, J., Forshaw, M., Stefanek, A., and Thomas, N. (2013). Time-inhomogeneous Population Models of a Cycle-Stealing Distributed System. UKPEW'13, pages 8–13.

Cappello, F., Geist, A., Gropp, B., Kale, L., Kramer, B., and Snir, M. (2009). Toward exascale resilience. *Int. J. High Perform. Comput. Appl.*, 23(4):374–388.

Choi, S., Baik, M., Hwang, C., Gil, J., and Yu, H. (2004). Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment. NCA '04, pages 366–371.

El Mehdi Diouri, M., Gluck, O., Lefevre, L., and Cappello, F. (2012). Energy considerations in checkpointing and fault tolerance protocols. DSN-W '12, pages 1–6.

Jarvis, S., Thomas, N., and van Moorsel, A. (2004). Open issues in grid performability. *IJSPM*, 5(5):3–12.

Li, J., Deshpande, A., Srinivasan, J., and Ma, X. (2009). Energy and performance impact of aggressive volunteer computing with multi-core computers. MASCOTS '09, pages 1–10.

Liang, S., Holmes, V., and Kureshi, I. (2012). Hybrid Computer Cluster with High Flexibility. CLUSTERW '12, pages 128–135.

Litzkow, M., Livney, M., and Mutka, M. W. (1998). Condor-a hunter of idle workstations. ICDCS '88, pages 104–111.

McGough, A., Gerrard, C., Noble, J., Robinson, P., and Wheater, S. (2011). Analysis of Power-Saving Techniques over a Large Multi-use Cluster. In *DASC'11*, pages 364–371.

McGough, A. S., Forshaw, M., Gerrard, C., Robinson, P., and Wheater, S. (2013). Analysis of power-saving techniques over a large multi-use cluster with variable workload. *CCPE*, 25(18):2501–2522.

Melhem, R., Mosse, D., and Elnozahy, E. (2004). The interplay of power management and fault recovery in real-time systems. *Computers*, 53(2):217–231.

Mills, B., Grant, R. E., Ferreira, K. B., and Riesen, R. (2013). Evaluating energy savings for checkpoint/restart. E2SC '13, pages 6:1–6:8. ACM.

Ren, X., Eigenmann, R., and Bagchi, S. (2007). Failure-aware Checkpointing in Fine-grained Cycle Sharing Systems. HPDC '07, pages 33–42. ACM.

Unsal, O. S., Koren, I., and Krishna, C. M. (2002). Towards energy-aware software-based fault tolerance in real-time systems. In *ISLPED*, pages 124–129.

UW-Madison (2013). UW-Madison CS Dept. HTCondor Pool Policies. http://research.cs.wisc.edu/htcondor/uwcs/policy.html.

Zhang, Y. and Chakrabarty, K. (2003). Energy-aware adaptive checkpointing in embedded real-time systems. In *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 918–923.